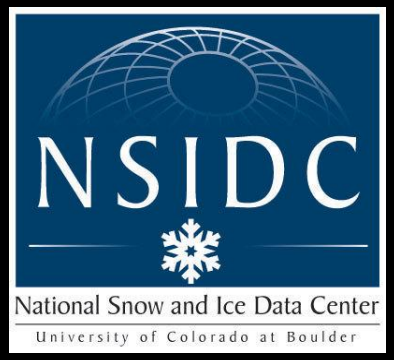
Strategies, motivations, and influencing adoption of testing for scientific code



Ian Truslove, Erik Jasiak

National Snow and Ice Data Center
University of Colorado Boulder

UCAR Software Engineering Assembly, 24 October 2013

Bios and Intros

Ian Truslove

Software Engineer
National Snow and Ice Data Center
@iantruslove
ian.truslove@nsidc.org

Erik Jasiak

Head of Software Development / Portfolio Manager
National Snow and Ice Data Center
@erikjasiak
erik.jasiak@nsidc.org

Slides: https://t.co/antAoj38tD

This presentation is licensed under a Creative Commons Attribution 3.0 Unported License

Aims

Make a case for using tests to improve valuable functional and non-functional attributes of scientific and research code.

Sketch out part of the spectrum of useful unit test related techniques.

Provide some social and organizational strategies to influence adoption.

Science Programming

Science Programming: A Second Class Citizen?

> *[...] recent studies have found that scientists typically spend 30% or more of their time developing software. However, 90% or more of them are primarily self-taught, and therefore lack exposure to basic software development practices [...]*

"Best Practices for Scientific Computing", Wilson et al, 2012
arXiv:1210.0530 [cs.MS]

Science Programming = Science

> *An article about computational science in a scientific publication isn't the scholarship itself, it's merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.*
>
> *- Jon Claerbout*

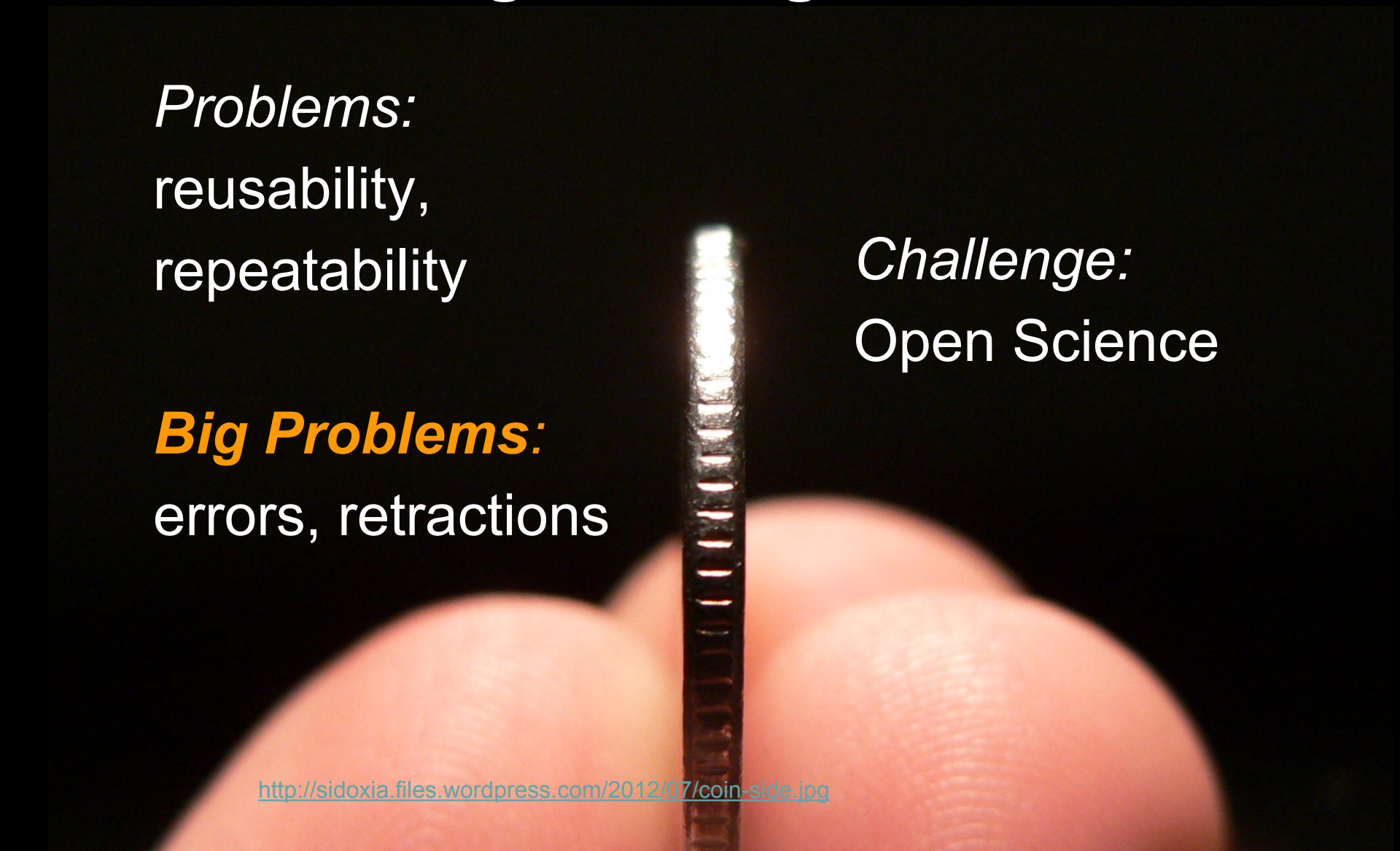
"The Legal Framework for Reproducible Scientific Research", Stodden, 2009

Science Programming

Problems:
reusability,
repeatability

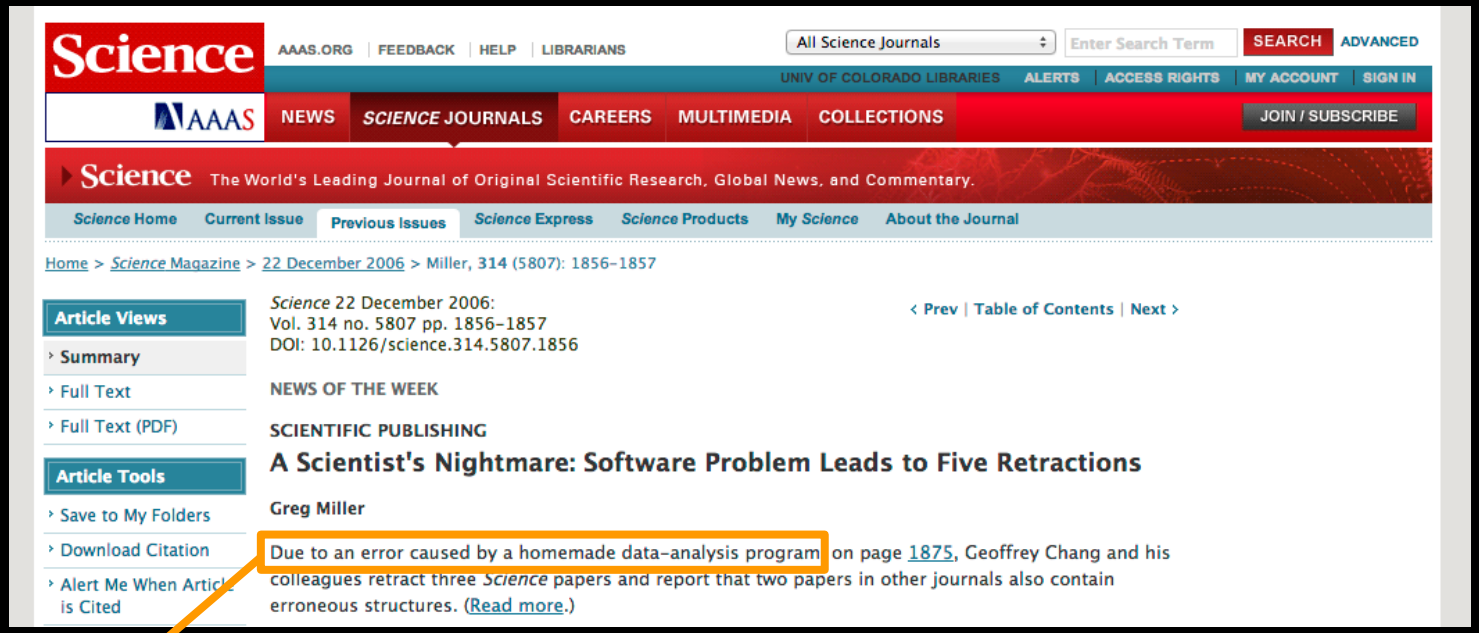
Big Problems:
errors, retractions

Challenge:
Open Science



http://sidoxia.files.wordpress.com/2012/07/coin-side.jpg

Your Nightmare?



"Due to an error caused by a *homemade* data-analysis program […]"

http://www.sciencemag.org/content/314/5807/1856

Towards Scientific Software Engineering

“Best Practices for Scientific Computing”

1. Write programs for people, not computers.
2. Automate repetitive tasks.
3. Use the computer to record history.
4. Make incremental changes.
5. Use version control.
6. Don’t repeat yourself (or others).
7. Plan for mistakes.
8. Optimize software only after it works correctly.
9. Document design and purpose, not mechanics.
10. Collaborate.

“Best Practices for Scientific Computing”, Wilson et al, 2012
arXiv:1210.0530 [cs.MS]

Proposed path for improvement: automated tests

Tests are executable specifications: repeatable, independent verification that the code is correct.

Proposed path for improvement: automated tests

Tests are executable specifications: repeatable, independent verification that the code is correct.

Huge spectrum of techniques: unit tests, test driven development, acceptance tests...

Not controversial: Beck, Feathers, Fowler, Hunt & Thomas (PragProg), Martin, Subramaniam, ...

Test Taxonomy

Unit tests primarily concerned with internal quality.

Integration tests, Acceptance tests concerned with external quality.

“Growing Object-Oriented Software, Guided by Tests”, Freeman & Pryce, 2010

About Unit Tests

Small piece of code to test another piece of code.

Expose the behavior of a particular *unit* of code (e.g. a method or function).

Typically written with a framework providing structural and assertion constructs.

Highlights regressions from the contract.

Good Unit Tests

FIRST:

- Fast
- Independent
- Repeatable
- Self-verifying
- Timely

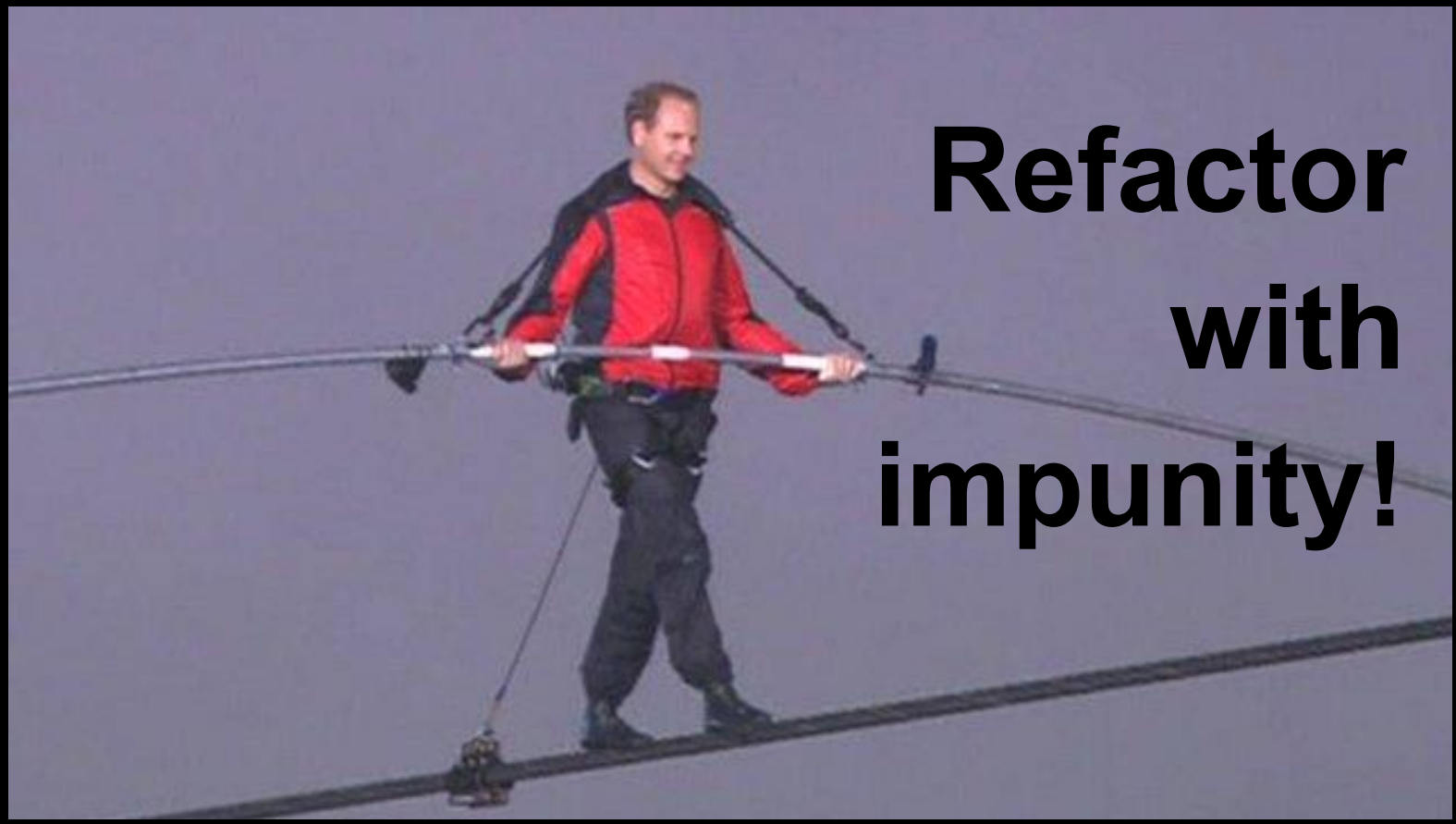
More on Good Unit Tests

Embody good software design

...but with an emphasis on readability.

http://codinghorror.typepad.com/.a/6a0120a85dcdae970b0128776fbe0b970c-pi

Fringe Benefits



http://news.bbcimg.co.uk/media/images/60949000/jpg/_60949195_jex_1437008_de36-1.jpg

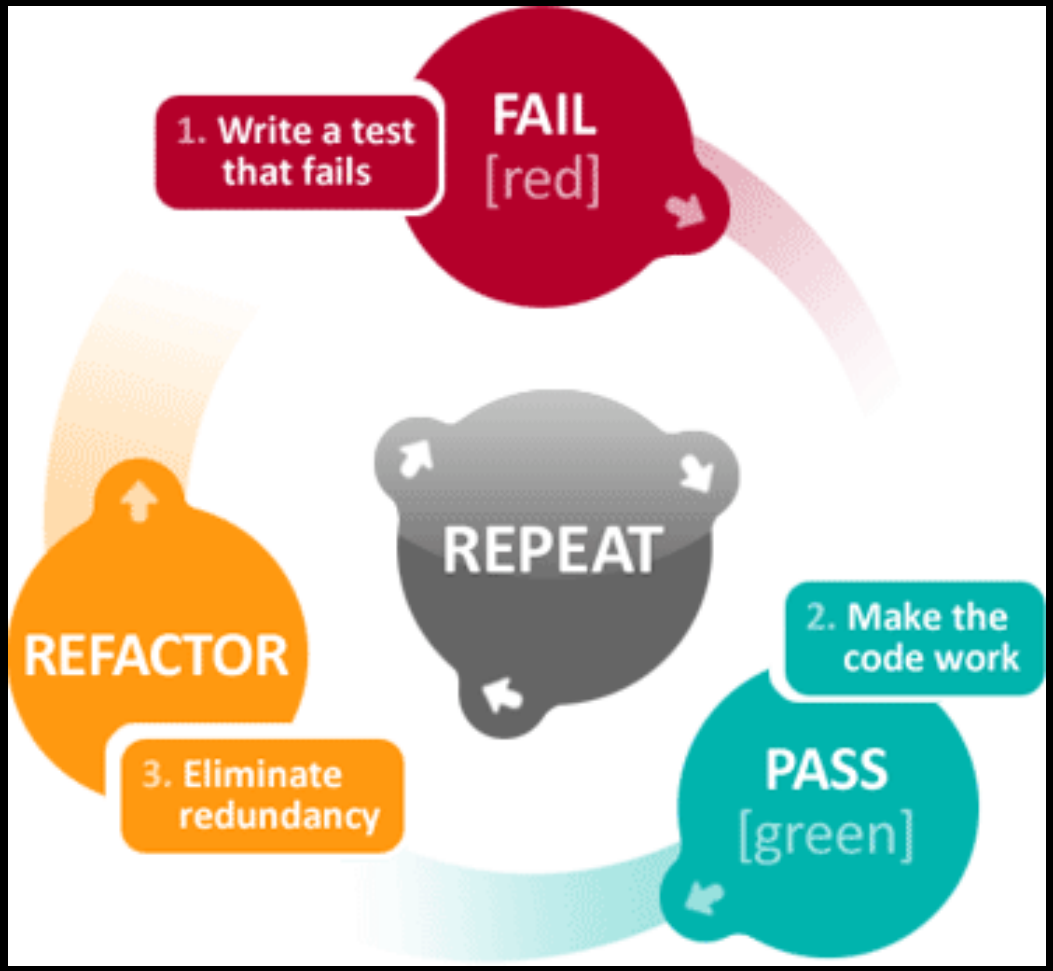
TDD - Test Driven Development

Write the tests for your code
before *writing the code itself*

TDD is a design exercise.

http://farm4.static.flickr.com/3087/2764892330_a8ffb86dcf.jpg

About TDD



http://www.agileapps.co.uk/images/methodology/tdd-red-green-refactor-diagram.gif

About TDD

- Clean code
  - it’s easy to make design or architectural changes
- Readable code
  - ...comes from the constant attention to detail
- Minimal code
  - no more than that which makes the tests pass
- Lots of tests
  - but no more than are needed

BDD - Behavior Driven Design

a "ubiquitous language" for analysis

Acceptance criteria should be executable

"Introducing BDD", Dan North

BDD: A language of executable specifications

```
1  require 'spec_helper'
2
3  describe 'Dataset Catalog Service App' do
4
5    it 'should provide dataset ISO xml at the dataset endpoint' do
6      get '/TEST_ID ', {}, { 'HTTP_ACCEPT' => 'application/x-iso19115+xml' }
7      last_response.should be_ok
8      last_response.header['Content-Type'].should match '^application/x\-iso19115\+xml'
9      last_response.body.should eql iso
10   end
11
12   it 'should provide an oai iso feed at the oai endpoint' do
13     get '/oai', { 'verb' => 'ListRecords', 'metadata_prefix' => 'iso' }
14     last_response.should be_ok
15     last_response.header['Content-Type'].should match '^application/xml'
16     last_response.body.should include(iso)
17   end
18 end
```

The Extended Test Family

Tests that aren’t unit tests:

- cover different scopes
- are still valuable
- shouldn’t be conflated with your unit tests!

Also consider stress tests, penetration tests, UI tests, ...

Acceptance Tests

End-to-end tests that verify the code *en bloc* meets the (user-facing) requirements.

Acceptance Tests: A domain-specific language for testing

```
Feature: We can apply a 3x3 neighborhood minimum filter to the data

  Scenario: Multiple minimums
    Given we have the following data layer:
      |   |   0 |   1 |   2 |   3 |   4 |
      | 0 | 100 | 100 | 100 | 100 | 100 |
      | 1 | 100 |  40 | 100 | 100 | 100 |
      | 2 | 100 | 100 |  50 | 100 | 100 |
      | 3 | 100 | 100 | 100 | 100 | 100 |
      | 4 | 100 | 100 | 100 | 100 | 100 |
    When we apply a neighborhood minimum filter
    Then the data should be:
      |   |   0 |   1 |   2 |   3 |   4 |
      | 0 |  40 |  40 |  40 | 100 | 100 |
      | 1 |  40 |  40 |  40 |  50 | 100 |
      | 2 |  40 |  40 |  40 |  50 | 100 |
      | 3 | 100 |  50 |  50 |  50 | 100 |
      | 4 | 100 | 100 | 100 | 100 | 100 |
```

Acceptance Tests: another DSL for a different domain

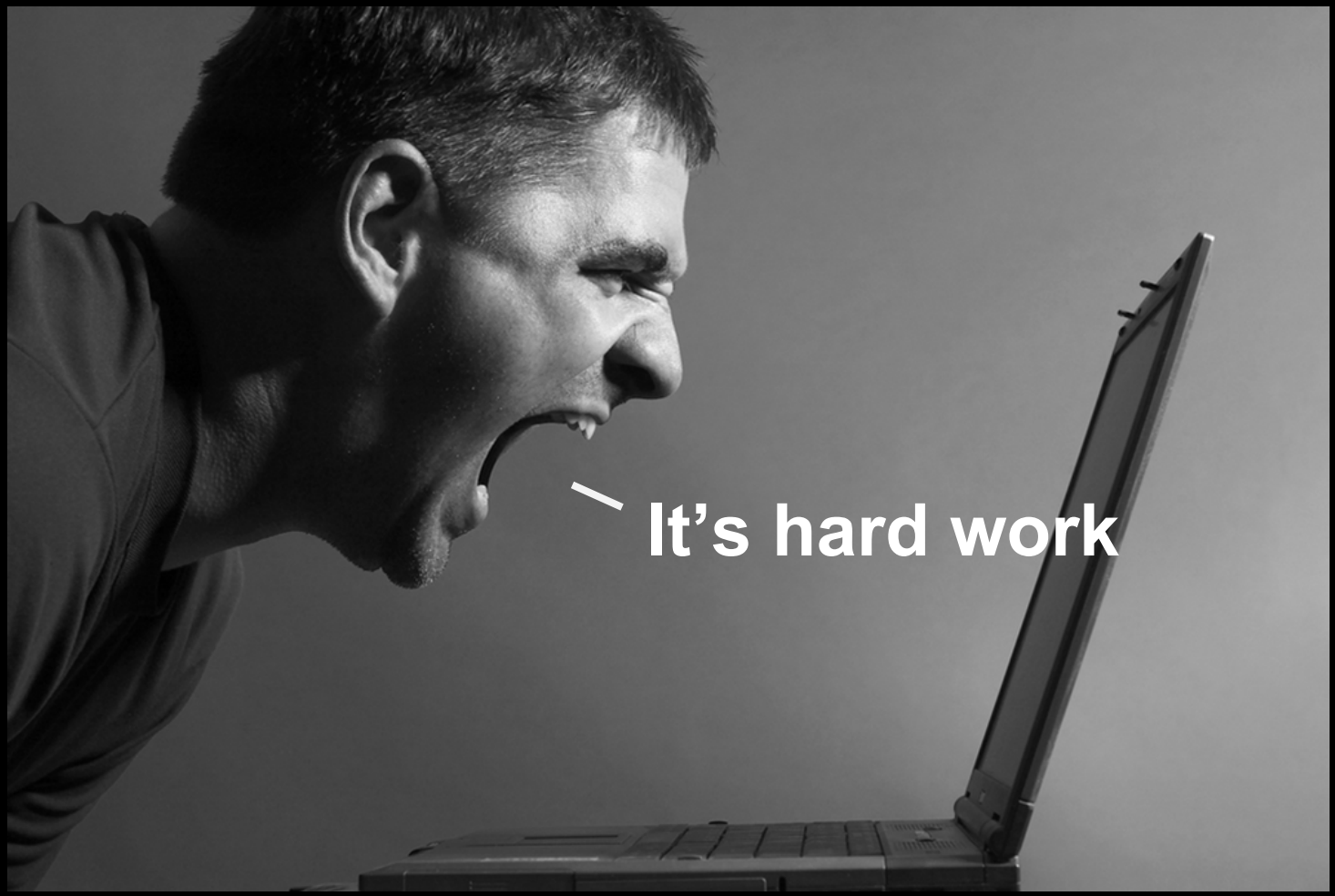
```
1  Feature: Basic ISO serialization of a single data set
2    Background:
3      Given there are the following valid environments:
4        | Environment | Hostname  | Port  | RelativeRoot    |
5        | development | localhost | 1580  |                 |
6        | production  | frozen    | 11580 | /api/dataset/2  |
7
8    Scenario: Successful response
9      When I request the ISO document for NSIDC-0051
10     Then I get a valid response
11     And the response has valid content
12
13   Scenario: Cache populated and used
14     When I clear the cache for NSIDC-0051
15     And I request the ISO document for NSIDC-0051
16     And I request the ISO document for NSIDC-0051
17     Then the second response is faster then the first response
```

Legacy Code

Legacy code = code without unit tests

...but how to get unit tests in?

Un-Legacy Code

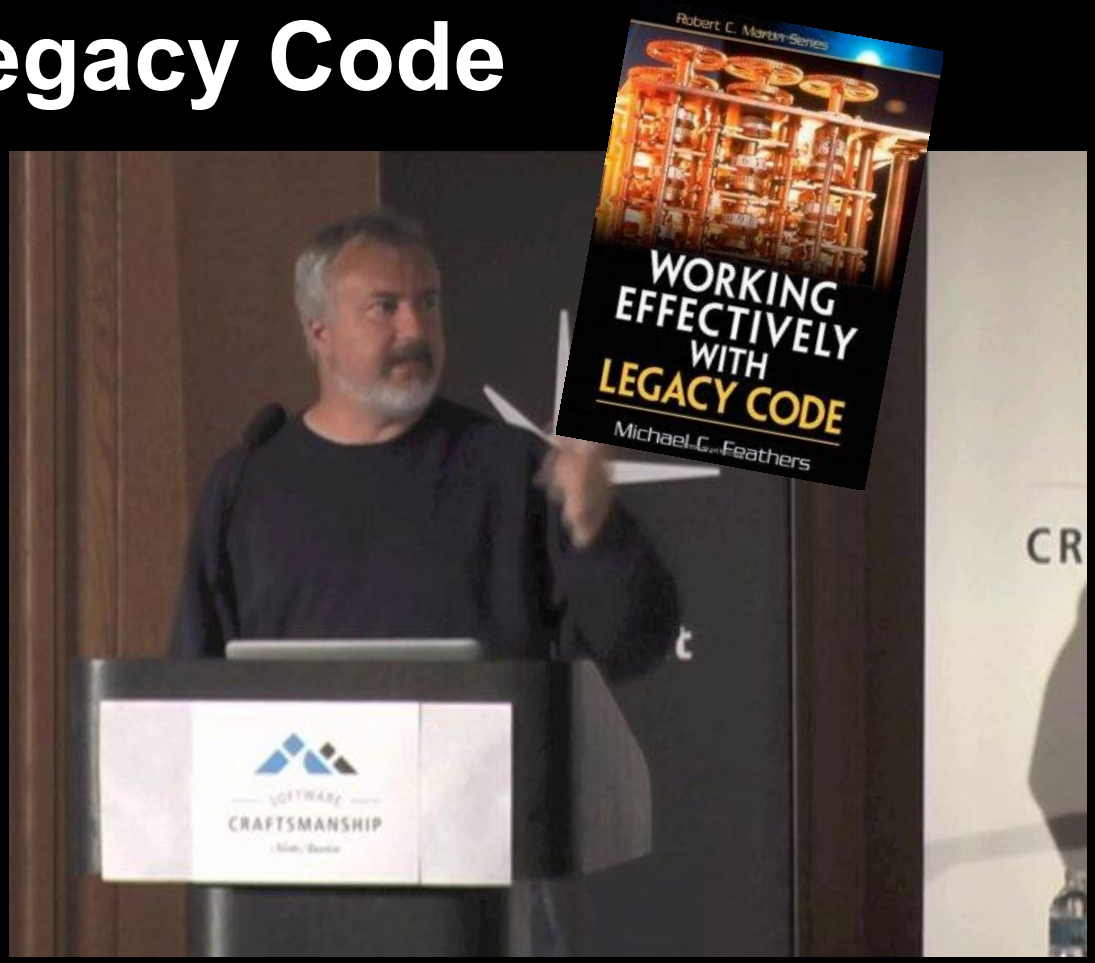


http://www.menspsychology.com/sites/default/files/facebook-frustration_full.png

Un-Legacy Code

http://worldonline.media.clients.ellingtoncms.com/img/croppedphotos/2011/04/18/trash_pick_up_day_t640.png?a6ea3ebd4438a44b86d2e9c39ecf7613005fe067

Un-Legacy Code



Feathers has great strategies

http://b.vimeocdn.com/ts/374/665/374665585_640.jpg

http://ecx.images-amazon.com/images/I/51S2-5n%2BJCL._SY300_.jpg

The Human Side

Cultural and Management Issues

Testing argument #427

http://dilbert.com/dyn/str_strip/000000000/00000000/0000000/000000/60000/5000/600/65675/65675.strip.zoom.gif (with edits)

“The Farmer and the Engineer”

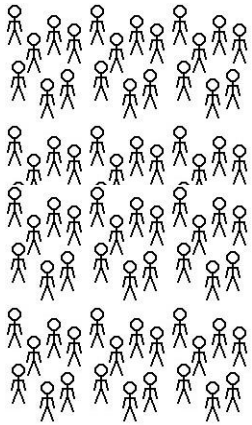
http://upload.wikimedia.org/wikipedia/commons/e/e7/Farmer_plowing_in_Fahrenwalde,_Mecklenburg-Vorpommern,_Germany.jpg

http://us.123rf.com/400wm/400/400/luislouro/luislouro0905/luislouro090500255/4824342-young-engineer-with-laptop-siting-in-a-tractor-isolated-against-a-white-background.jpg

We are all matrixed

Science Goals

Development goals

	 	 	
One Person	Two People	Team	Organization

* image citations at end

The Arguments Against ~~Testing~~ Anything

"This is going to take longer!"

- Nefarious counterpart: "Time writing X lines of code" as a measurement.

"I'm opposed to process mandates."

"Nobody cares about my code but me..."

- "...and I'm never going to show anyone anyway."

Complexity in Scientific Software

$$\mathcal{F}f[m] = \mathcal{F}f_{\text{even}}[m] + \omega[n, -m]\mathcal{F}f_{\text{odd}}[m]$$

$$\mathcal{F}f[m + n/2] = \mathcal{F}f_{\text{even}}[m] - \omega[n, -m]\mathcal{F}f_{\text{odd}}[m]$$

```
1  def fft(signal):
2     n = len(signal)
3     if n == 1:
4        return signal
5     else:
6        Feven = fft([signal[i] for i in xrange(0, n, 2)])
7        Fodd = fft([signal[i] for i in xrange(1, n, 2)])
8
9        combined = [0] * n
10       for m in xrange(n/2):
11          combined[m] = Feven[m] + omega(n, -m) * Fodd[m]
12          combined[m + n/2] = Feven[m] - omega(n, -m) * Fodd[m]
13
14       return combined
```

http://jeremykun.com/2012/07/18/the-fast-fourier-transform/

The myth of lines of code

- Award winning “best lines of code ever”
- Runs in perl **and** postscript!

The myth of lines of code

- Award winning “best lines of code ever”
- Runs in perl **and** postscript!

```
/;{}def/#{def}def/$_={/Times-Bold exch selectfont}#/_{rmoveto}#/"{dup}#/*/!/$
;/q{exch}#/x ; {/J q #}#/.{/T q #}#{stringwidth}#{}#{}# 14 string dup dup dup
260 40 moveto 90 rotate ; %/}};$0='"\e[7m \e[0m"';@ARGV=split//,reverse
q(ThePerl). q(Journal) x 220 ; q ; 0 T putinterval exch 7 J putinterval ;
 ; $_= q /m$ pop T($*!$"=!$ " )pop " * true% ? $ " $!" "  !!  !! % !" !"    !
! charpath {!"""}pop $ pop{""!}pop ! neg{!#}pop 220 ! neg _{!!}pop J false %T
charpath  clip " pop 0 " moveto 6{!!}pop $_= 105{!!}pop {$ ! $ " !  #! ##}
pop{dup dup $ ! " pop pop q{"}pop 22{dup show}repeat {"}pop q 22 mul{$ "} pop
neg{!#! $ "}pop ! 8 .65 mul{$ # # $}pop ! neg{"}pop  _ pop{"}pop } repeat pop
" {  $ " ! ! ! $ " ! !" "#"  #"!""""""! #" " # "m/;@ARGV=(@ARGV[-14..-1])x50;q}
 0 "%};s/m[ou]|[-\dA-In-z.\n_{}]|\$_=//gx;s/(.)(?{$*="})/('$*.='.(++$#
%2?":"$0;").'pop;')x(ord($1)-31).'$*'/gee;s/((.(\e\[.m)*|.){77})/$1\n/g;print
; sub showpage {}
```

The myth of lines of code - pt 2

Development effort is not measured in keystrokes.

How much brainpower does my code consume for everyone involved?

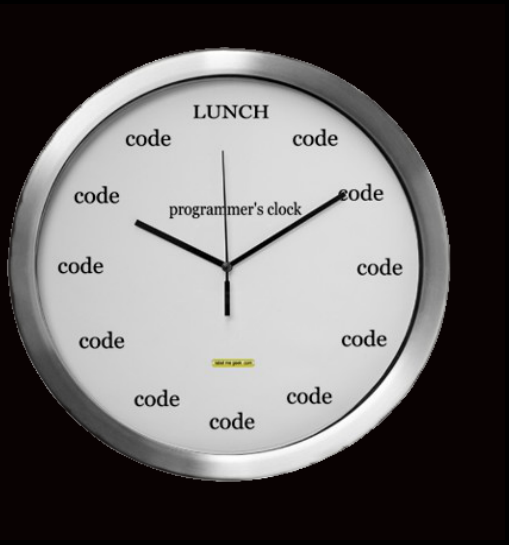
```
if (string.IsNullOrEmpty(filetext) || !filetext.Contains(",")) goto end;
```

```
*++b ? (*++b + *(b-1)) 0
```

```
getElements( ‘p’, {'class':'statusbar'} )[0].firstChild.innerText
```

The reality of lines of code

The real point: how much time does my code take to create and debug and maintain?



http://www.photoshop3d.com/wp-content/uploads/2007/02/hourglass_00.jpg
http://i1.cpcache.com/product/382140623/programmers_modern_wall_clock.jpg?color=Silver&height=460&width=460&qv=90
http://cdn.homedit.com/wp-content/uploads/2011/02/hwc1.jpg

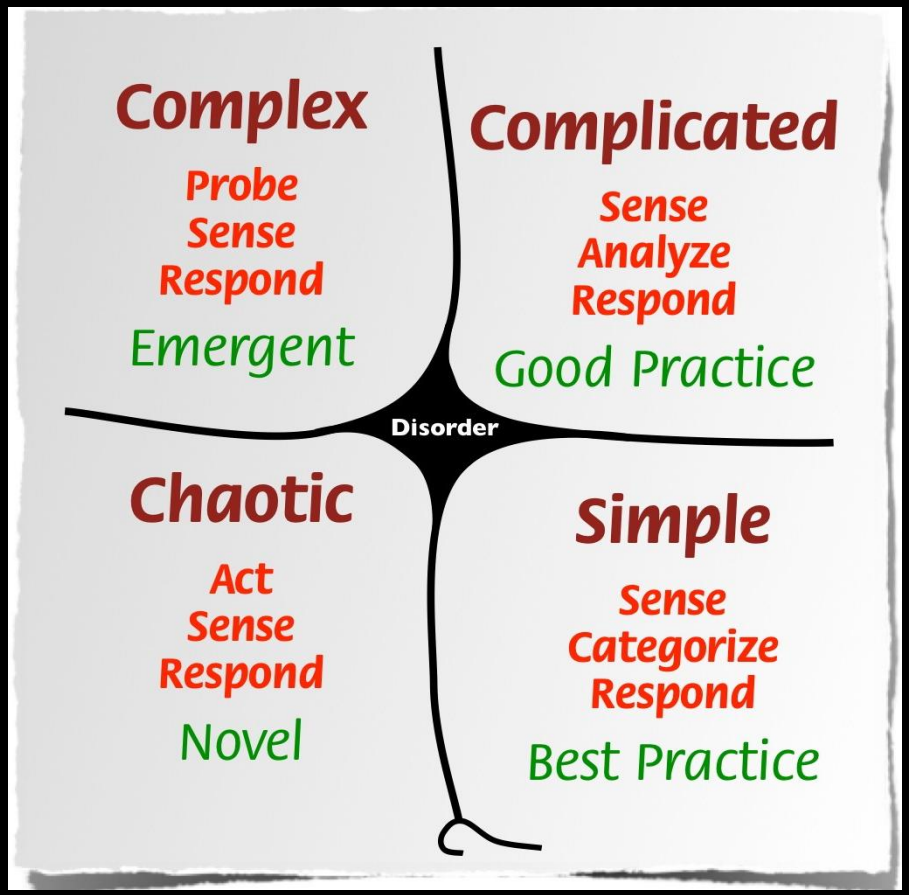
When is our fft accepted “as given,” and the focus on its application?

```
1  import unittest
2
3  class CoreFFTTests(unittest.TestCase)
4    def realNumberTest(self):
5       # [Test some real numbers]
6
7    def imaginaryNumberTest(self):
8       # [Test some imaginary numbers]
9
10   def complexNumberTests(self):
11      # [Test some complex numbers]
```

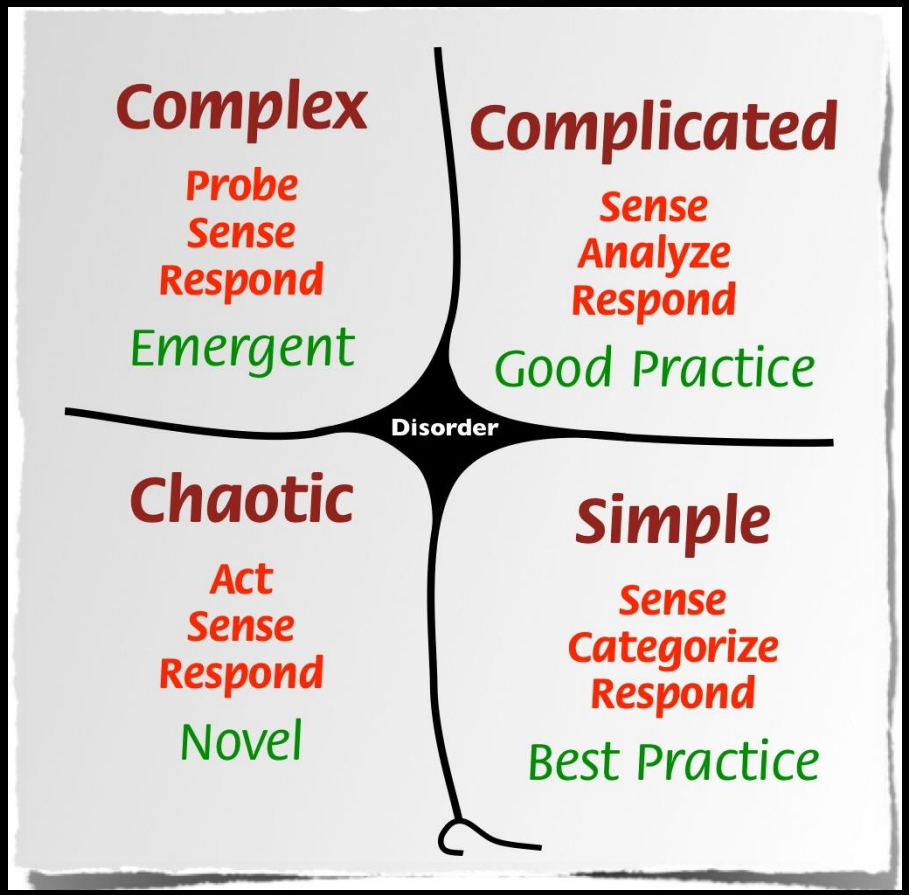
```
1  def fft(signal):
2     n = len(signal)
3     if n == 1:
4        return signal
5     else:
6        Feven = fft([signal[i] for i in xrange(0, n, 2)])
7        Fodd = fft([signal[i] for i in xrange(1, n, 2)])
8
9        combined = [0] * n
10       for m in xrange(n/2):
11          combined[m] = Feven[m] + omega(n, -m) * Fodd[m]
12          combined[m + n/2] = Feven[m] - omega(n, -m) * Fodd[m]
13
14       return combined
```

http://jeremykun.com/2012/07/18/the-fast-fourier-transform/

Process vs good practice



Process vs good practice



Software Engineering

Operational Procedure (Process)

http://en.wikipedia.org/wiki/File:Cynefin_framework_Feb_2011.jpeg

Serving the World Wide Web - 1990

https://github.com/NotTheRealTimBL/WWWDaemon/tree/master/old

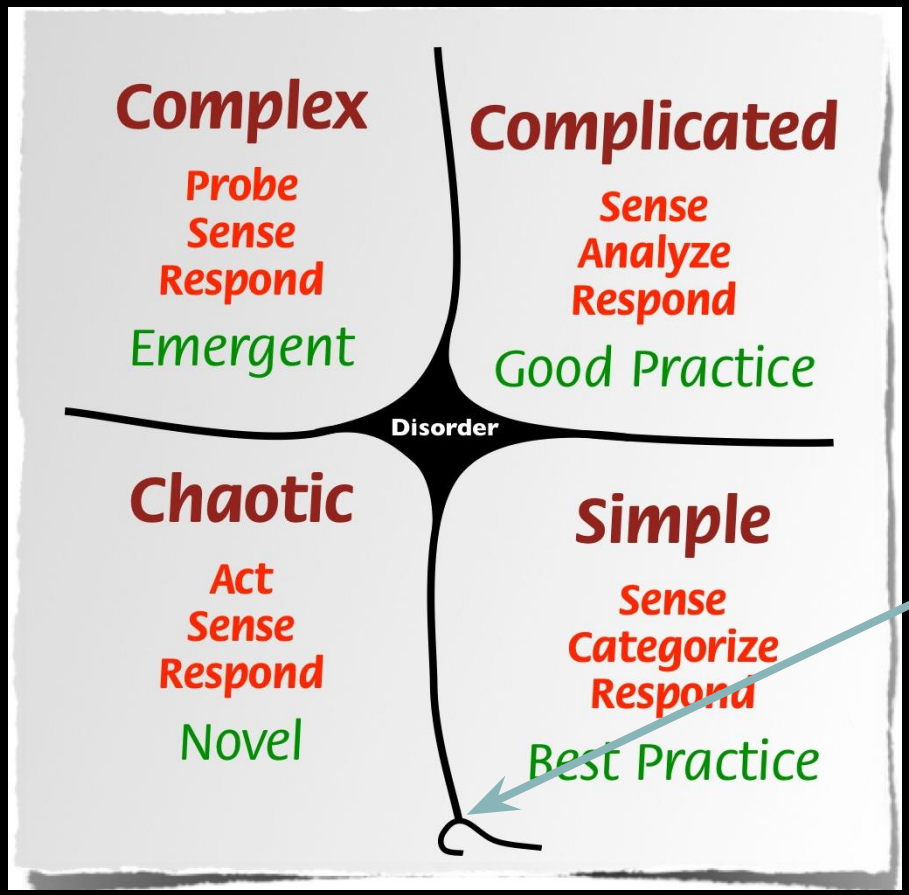
http://www.wimklerkx.nl/IN/IN22-14.jpg

Complex -> complicated (but not simple)

```
 1 import sys
 2 import BaseHTTPServer
 3 from SimpleHTTPServer import SimpleHTTPRequestHandler
 4
 5 HandlerClass = SimpleHTTPRequestHandler
 6 ServerClass  = BaseHTTPServer
 7 Protocol  = "HTTP/1.0"
 8
 9 if sys.argv[1:]:
10    port = int(sys.argv[1])
11 else
12    port = 8000
13
14 server_address = ('127.0.0.1', port)
15 HandlerClass.protocol_version = Protocol
16
17 # A MIRACLE!
18 httpd = ServerClass(server_address, HandlerClass)
19
20 sa = httpd.socket.getsockname()
21 print "Serving HTTP on"  sa[0]  "port" sa[1]  "..."
22 httpd.serve_forever()
```

http://www.linuxjournal.com/content/tech-tip-really-simple-http-server-python

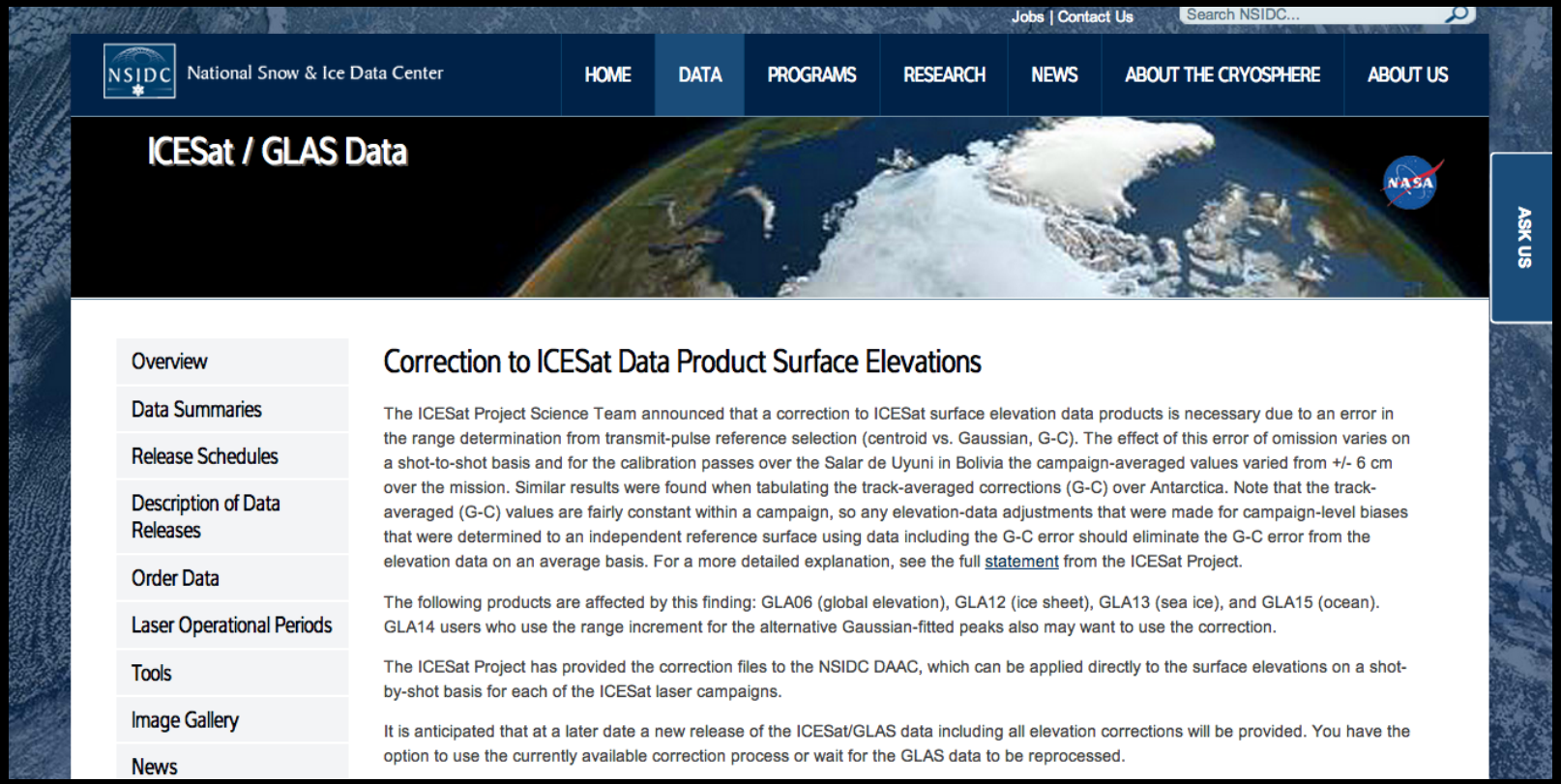
Process vs good practice



The Complexity Cliff (or, why reusing old Fortran will kill you someday.)

http://en.wikipedia.org/wiki/File:Cynefin_framework_Feb_2011.jpeg

My code: mine alone.



My code: mine alone?

> *Data is a precious thing and will last longer than the systems themselves.*
>
> *Tim Berners-Lee*

- Who owns the results?
- How long will this be around?
- Will your work ever need to be verified?
- Will the data you produce ever need to be expanded upon?

The risk is real
- but the risk is not persuasive!

Google “persuasion” -

- then tell me how often negative methods work

The risk is real
- but the risk is not persuasive!

Be positive (it's more than just a blood type)

- Humor
- Create a positive experience
- Appeal to order
- Appeal to senses
- Endorsement
- Bandwagon
- [...]

Nuts and bolts technique:

Stroke the (research) ego.

http://images4.wikia.nocookie.net/__cb20130418031554/poohadventures/images/5/5e/Anton_Ego.jpg

Repeat our confidence test

```
1  import unittest
2
3  class CoreFFTTests(unittest.TestCase)
4    def realNumberTest(self):
5       # [Test some real numbers]
6
7    def imaginaryNumberTest(self):
8       # [Test some imaginary numbers]
9
10   def complexNumberTests(self):
11      # [Test some complex numbers]
```

```
1  def fft(signal):
2     n = len(signal)
3     if n == 1:
4        return signal
5     else:
6        Feven = fft([signal[i] for i in xrange(0, n, 2)])
7        Fodd = fft([signal[i] for i in xrange(1, n, 2)])
8
9        combined = [0] * n
10       for m in xrange(n/2):
11          combined[m] = Feven[m] + omega(n, -m) * Fodd[m]
12          combined[m + n/2] = Feven[m] - omega(n, -m) * Fodd[m]
13
14       return combined
```

http://jeremykun.com/2012/07/18/the-fast-fourier-transform/

Nuts and bolts technique:

Stroke the (research) ego.

“More people will use this!”

- More citations
- More exposure
- More trust

http://images4.wikia.nocookie.net/__cb20130418031554/poohadventures/images/5/5e/Anton_Ego.jpg

Summary: Strategies

- More lines of code: change the argument.
  - It's about time, maintenance, efficiency
- Opposition to process / mandate: change the argument again.
  - We're looking to mitigate complexity with good practice, not force another barrier.
- It's my code; nobody else cares.
  - Carrot: Naked, unbridled flattery of their work.
  - (Stick: call from Bill O'Reilly if errors are found)

What you can do to help

(Hint: don't do this.)

http://robwellsblog.files.wordpress.com/2013/08/sci-behind-testing.jpg?w=696

What you can do to help

(Hint: or this.)

http://blog.decayingcode.com/image.axd?picture=mycodecantfail_thumb.jpg

Summary of what we do at NSIDC

BDD and TDD in Python, Ruby, JavaScript, Java.

Acceptance testing, including web interfaces.

Continuous integration - all tests are run on every check in.

Continuous delivery: push-button releases.

Further Reading

Anne Wilson (LASP) is also to thank for some of these…

Scholarly articles:

- Wilson et al, "Best Practices for Scientific Computing", highly recommended.
- Merali, "Computational science: ... Error" in Nature. *"As a general rule, researchers do not test or document their programs rigorously, and they rarely release their codes, making it almost impossible to reproduce and verity published results generated by scientific software, say computer scientists."*

Good books for unit testing, TDD, and higher-level tests:

- Freeman and Pryce, Growing Object-Oriented Software, Guided by Tests
- Beck, Test Driven Development: By Example
- Fowler, Mocks Aren't Stubs - Martin Fowler on the terminology and usage of mocks, stubs, test doubles - all those "fake collaborators"
- A couple of Bob Martin's books are particularly noteworthy for covering lots and lots of desirable attributes for code
- Martin, Agile Software Development, Principles, Patterns, and Practices
- Martin, Clean Code: A Handbook of Agile Software Craftsmanship

Also:

- Feathers, Michael C., Working Effectively with Legacy Code , Prentice Hall, 2005, p. xvi.
- Snowden, Cynefin: Wikipedia on Cynefin, David Snowden introducing Cynefin (video) - applicable to knowledge management, cultural change, and community dynamics, and has also involved issues of organizational strategy.
- Snowden, Boone, 2007 "A Leader's Framework for Decision Making" (must pay for access from Harvard Business Review, though perhaps available elsewhere)

Some test frameworks

JUnit - the original (Java)
RSpec (Ruby)
Behave (Python)
Jasmine (JavaScript)
mgunit (IDL)
pfUnit (FORTRAN)
Cucumber (Acceptance tests, lots of languages)

Additional image credits (matrix org slide)

www.simtalk.com/news-2-you/Penguin/Page10.htm

http://images5.cafepress.com/image/31972305_150x150.png

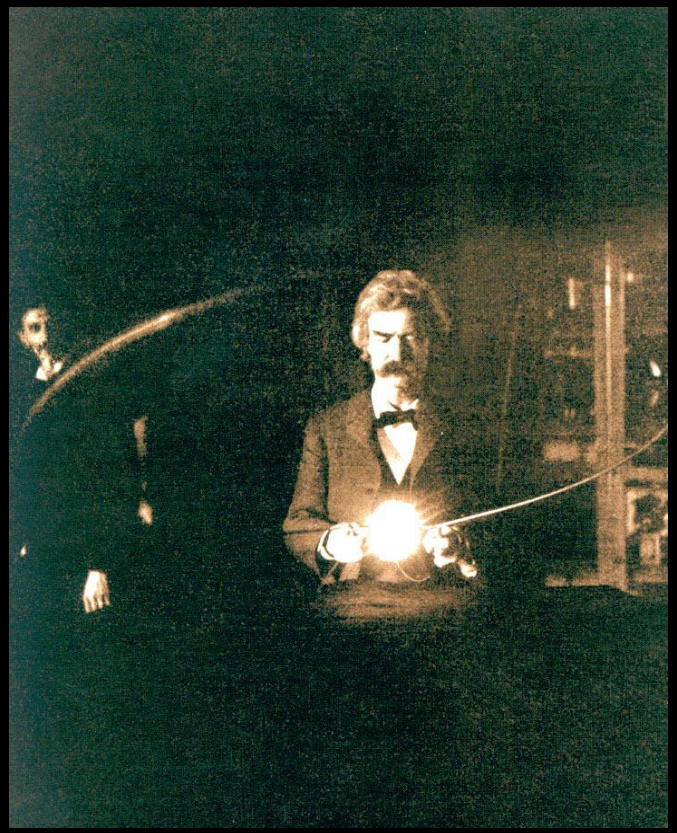
http://uwcpsl.files.wordpress.com/2013/02/teamwork.png

http://morganbeebe.files.wordpress.com/2011/07/stick-figure-group-of-63.jpg

Thank you!

@iantruslove
ian.truslove@nsidc.org

@erikjasiak
erik.jasiak@nsidc.org



Slides: https://t.co/antAoj38tD

http://www.teslasociety.com/pictures/ntesla4.jpg